Affichage interactif, bidimensionnel et incrémental de formules mathématiques

Hanane Naciri et Laurence Rideau
INRIA Sophia Antipolis, BP. 93, 06902 Sophia Antipolis Cedex, France.
{Hanane.Naciri,Laurence.Rideau}@sophia.inria.fr

Mots clés: interaction homme machine, outils de formatage, affichage de formules mathématiques.

Résumé: FIGUE est un moteur d'affichage interactif incrémental et bidimensionnel, développé à l'INRIA. FIGUE et son extension AÏOLI (bibliothèque dédiée à la manipulation d'objets structurés) constituent une boîte à outils pour le développement d'interfaces graphiques et d'éditeurs d'objets structurés. Dans cet article, nous décrivons comment utiliser et étendre FIGUE pour afficher des formules mathématiques et pour interagir avec ces formules.

Abstract: FIGUE is an incremental two dimensional layout engine, developped at INRIA. Used with AÏOLI (a toolkit to build interactive applications that manipulate data as trees) it becomes a specialized toolbox for building customized editors. In this paper we describe how to use and extend FIGUE to display mathematics formulas and interact with them.

1 Introduction

La conception d'outils de formatage génériques, portables et communicant facilement avec des outils d'aide au développement d'interfaces homme-machine présente un grand intérêt. Elle participe à améliorer la qualité de ces interfaces. De tels outils de formatage doivent résoudre le problème lié à la diversité des objets à manipuler (du texte simple, des formules mathématiques pour les systèmes de preuves ou de calcul formel, des images,...). Des problèmes particuliers se posent pour les formules mathématiques qui ont une structure complexe et bidimensionnelle (matrices, intégrales,...).

Nous distinguons deux familles d'outils de formatage de documents :

- les outils d'édition de documents en mode non-iteractif (*batch-oriented processors*), comme par exemple le système TEX [12], qui permet un formatage de grande qualité;
- les éditeurs WYSIWYG (*What You See Is What You Get*), comme par exemple AMAYA [17] (éditeur WYSIWYG de pages Web contenant des formules mathématiques).

Ces derniers éditeurs offrent un environnement graphique qui aide l'utilisateur à visualiser son document pendant toute la phase de création. Ils permettent de manipuler dynamiquement des objets graphiques contrairement aux outils de la première famille qui nécessitent la transcription du document à l'aide d'un langage qui sera interprété ultérieurement pour l'affichage.

FIGUE est un module de formatage et d'affichage 2D portable utilisé dans le développement d'outils WYSIWYG. Il est utilisé aujourd'hui dans l'équipe LEMME [1] de l'INRIA Sophia-Antipolis pour le développement d'outils interactifs et d'éditeurs d'objets structurés comme les programmes ou les formules mathématiques.

1. Logiciels et Mathématiques

Au départ, une version préliminaire de FIGUE, intégrée au système CENTAUR [6], a été utilisée pour le développement de CAS/PI [9], une interface utilisateur générique pour les systèmes du calcul formel. Actuellement, FIGUE (réécrit totalement en JAVA) et son extension AÏOLI [15](bibliothèque dédiée à la manipulation d'objets structurés) sont utilisés pour le développement du système PCOQ, une interface graphique conçue pour le système de développement de preuves COQ [7]. A terme, il est envisagé de lier FIGUE aux outils d'affichage d'objets structurés pour le Web dans le genre de AMAYA (décrit ci-dessus). FIGUE aura le rôle de passerelle entre les interfaces des applications et le monde Web.

Dans la section suivante, l'architecture du système PCOQ sera détaillée afin d'illustrer le rôle et les fonctionnalités principales de FIGUE. Ensuite, les caractéristiques de FIGUE pour manipuler des objets structurés seront expliquées en détail. Nous insisterons sur l'extensibilité de FIGUE qui nous a servi en particulier pour l'affichage des formules mathématiques.

2 PCOQ : un exemple d'utilisation de FIGUE

PCOQ (la version JAVA de CTCOQ [3] en cours de développement) fournit un environnement de travail graphique et interactif pour le système de preuve COQ [7]. Le moteur de preuve et l'interface utilisateur sont deux processus indépendants. Le but principal de PCOQ est de supporter le développement de preuves à grande échelle et de fournir une interface conviviale facilitant la création de preuves pour les utilisateurs du système de preuve COQ. Cette interface possède les caractéristiques suivantes :

- une interface graphique : présentation structurée et colorée des formules et des commandes ;
- des mécanismes d'édition et de présentation structurées : l'environnement fournit les moyens d'éditer structurellement formules et commandes ;
- la preuve par sélection : l'environnement utilise la structure des formules logiques pour aider systématiquement l'utilisateur à guider la preuve par de simples sélections à la souris [4].

L'interface graphique se base sur la boîte à outils AÏOLI et le noyau d'affichage FIGUE. Les données venant de COQ sont transformées en une structure d'arbres. Ensuite, AÏOLI manipule cette structure en utilisant ses trois composantes principales, inspirées des concepts du système CENTAUR [6] :

- VTP (*Virtual Tree Processor*) : une machine abstraite permettant de spécifier et de manipuler des structures d'arbres en suivant un formalisme (définition par des types abstraits d'un langage) ;
- PPML (*Pretty Printing Meta Language*): un langage permettant de décompiler un arbre de syntaxe abstraite en un arbre de boîtes. Une spécification PPML est une suite de règles où chaque règle se compose d'une partie gauche représentant un arbre de syntaxe abstraite et une partie droite décrivant le formatage correspondant [8] ;
- GFXOBJ: une librairie d'objets graphiques (menus, fenêtres, etc ...) permettant le développement d'applications interactives pour les arbres VTP.

AÏOLI, à l'aide de PPML, traduit un arbre de syntaxe abstraite en un arbre de boîtes. A partir de cet arbre de boîtes, FIGUE construit l'arbre de boîtes formaté qui sera affiché par la suite.

La figure 1 montre un exemple de session PCOQ (preuve en COQ que $\sqrt{2}$ n'est pas rationnel: $\sqrt{2} \notin \mathbb{Q}$). Dans cet exemple, différentes notations sont manipulées : texte, formules mathématiques et images. L'utilisateur peut effectuer des preuves par sélection [4], i.e. la sélection d'une formule logique permet la génération automatique de la commande à envoyer à COQ (en fonction du contexte, de la position de l'expression sélectionnée, ...). PCOQ offre aussi la possibilité de sélectionner et de déplacer un terme d'une formule (*drag and drop* [5]) en appliquant la transformation correspondante. Si nous prenons l'exemple simple de l'équation $ax + b = 0$ et que nous déplaçons le paramètre b du côté droit, nous obtenons l'équation $ax = -b$.

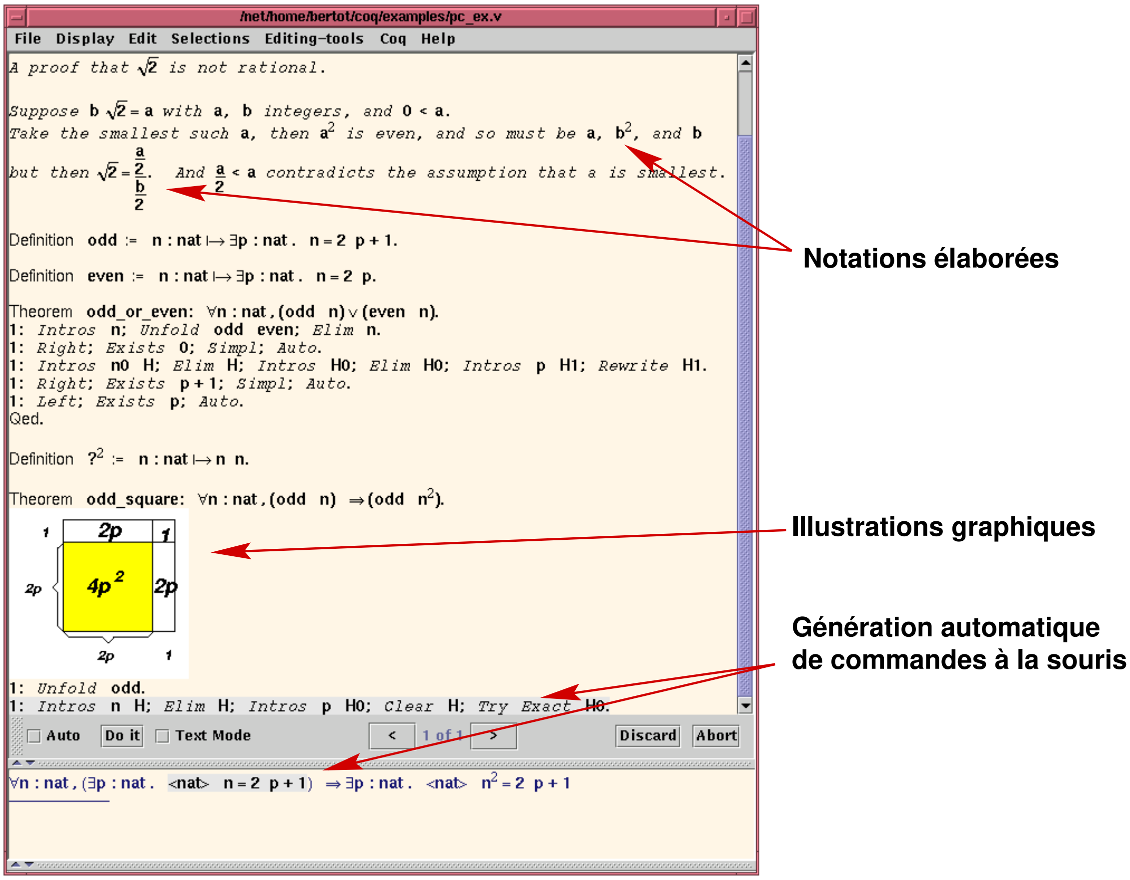


FIG. 1 – *Une session* PCOQ.

3 Affichage et manipulation d'objets structurés à l'aide de FIGUE

FIGUE est un moteur d'affichage bidimensionnel, incrémental, entièrement écrit en langage JAVA. Dans cette section, nous présentons ses principales caractéristiques : les structures de boîtes, les algorithmes de formatage et d'affichage, les possibilités d'incrémentalité et de gestion de sélection à la souris, la notion de constructeurs et son extensibilité pour l'affichage de formules mathématiques.

3.1 Structures de boîtes

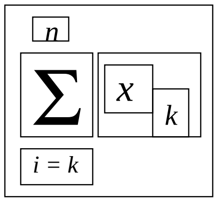
FIGUE se base sur la notion de boîtes [12] pour représenter tous les objets structurés du document. Il manipule un arbre de boîtes produit par les règles du langage PPML. Prenons un exemple simple d'une règle PPML pour l'opérateur binaire d'addition :

$$\text{plus}(*x, *y) \quad \rightarrow \quad [<\text{hv } 1,1,0> *x \text{ "+" } *y]$$

On associe à un arbre de syntaxe abstraite dont la racine est un “plus”, une boîte composée d'un constructeur graphique *hv* (voir section 3.5) et trois éléments $*x$, $+$ et $*y$. En partie droite de la règle, $*x$ et $*y$ sont des appels récursifs de l'affichage sur les fils de l'opérateur plus. Ils seront séparés par la chaîne "+" qui est un atome (une feuille de l'arbre de boîtes). Le constructeur *hv* permet de formater en paragraphe ses fils avec différents paramètres (indentation, espacement, interligne).

Les objets du document sont représentés sous forme d'arbre de boîtes, où chaque nœud est une boîte englobant toutes ses boîtes filles et les feuilles sont des atomes. La figure 2 montre un exemple simple de représentation en boîtes d'une formule mathématique.

$$\sum_{i=k}^{n} x_k \qquad \xrightarrow[ppml]{}$$



(a) (b)

FIG. 2 – *Transformation d'une formule mathématique (a) en un ensemble de boîtes (b), à l'aide de règles* PPML.

3.2 Formatage et affichage bidimensionnels

Le formatage et l'affichage au niveau de FIGUE sont bidimensionnels, par opposition aux méthodes employées par certaines interfaces qui obligent à linéariser les notations. Ils correspondent aux notations employées pour les documents manuscrits ou typographiés.

L'algorithme de formatage prend en compte la largeur de la page (pour que les boîtes soient visibles), les multiples polices utilisées et les différents paramètres liés au contexte de chaque boîte. Il travaille de façon incrémentale, i.e. quand une boîte est modifiée, il ne recalcule que le positionnement des boîtes concernées. Le formatage consiste à parcourir l'arbre de boîtes en mettant à jour leurs attributs : taille, positionnement (origine), alignement (points d'entrée et de sortie). Une fois le formatage effectué, l'algorithme d'affichage parcourt simplement l'arbre de boîtes pour afficher les boîtes en fonction de leur contexte graphique (police de caractères, couleur, coordonnées).

3.3 Incrémentalité

En mode interactif, nous devons minimiser le coût de reformatage dû à la mise à jour ou à la sélection d'une ou plusieurs boîtes. Chaque modification d'une boîte (contexte, taille, origine,. . .) doit être propagée dans l'ensemble de l'arbre de façon optimale afin de mettre à jour seulement les boîtes affectées. Dans la version actuelle de FIGUE, l'algorithme de formatage incrémental est plutôt simple. En mettant à jour la structure de l'arbre de boîtes, les modifications sont propagées vers la racine de père en père. Cet algorithme utilise les propriétés particulières de chaque boîte. Par exemple, dans le cas d'une boîte qui dispose ses fils horizontalement, nous ne reformatons qu'à partir du premier fils modifié (les autres fils seront poussés vers la droite, voir figure 3). Dans FIGUE, il est possible de spécialiser l'algorithme de reformatage pour chaque boîte.

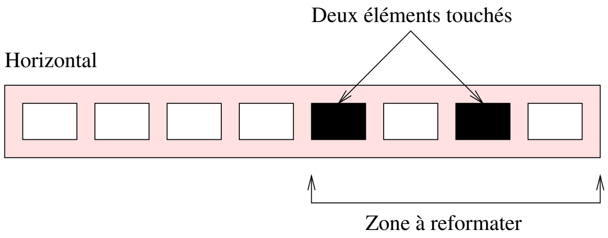


FIG. 3 – *Reformatage incrémental de la boîte Horizontal.*

3.4 Sélection

La plupart des interfaces modernes et interactives offrent la possibilité de sélectionner à la souris les objets affichés afin de les manipuler dynamiquement (par exemple, copier-coller). Le mécanisme de sélection de FIGUE est directement lié à la structure de l'arbre de boîtes. Il traduit automatiquement les coordonnées de la souris en un emplacement dans la structure des boîtes. La sélection peut être simple (un seul élément) ou multiple (plusieurs éléments sélectionnés à la fois). Lorsque une boîte est sélectionnée, une opération inverse de l'affichage (voir figure 2) permet d'obtenir le sous-arbre de la structure syntaxique sous-jacente correspondant à cette boîte.[2]

Dans le cas d'une sélection multiple de boîtes, il n'y a pas forcément un sous-arbre directement associé à cette sélection; nous pouvons néammoins formuler des heuristiques qui permettent une sélection multiple structurée. Par exemple, dans PCOQ on détermine la plus petite sous-expression contenant tous les objets sélectionnés. Si on prend l'expression mathématique $(a*c)+b$ et qu'on sélectionne a et c, la sous-expression $a*c$ sera sélectionnée. Par contre, si on sélectionne c et b alors l'expression $(a*c)+b$ sera sélectionnée entièrement. Malheureusement, dans le cas où l'on veut sélectionner une diagonale d'une matrice, cette méthode sélectionne toute la matrice (la plus petite structure contenant tous les éléments sélectionnés). Toutefois, il est possible d'affiner la méthode de sélection et de construire une sous-liste "virtuelle" des éléments de la diagonale sur laquelle l'interface pourra travailler comme si cette sous-liste existait vraiment dans l'objet structuré sous-jacent.

La sélection est une fonctionnalité très importante et trés utile dans les interfaces homme-machine. PCOQ offre un mécanisme puissant de sélection : comme par exemple des preuves par sélection et du *drag and drop* décrits dans la section 2.

2. on détermine la règle PPML qui est à l'origine de l'affichage de cette boîte. Le sous-arbre sélectionné correspond alors au pattern de la partie gauche de la règle PPML

3.5 Constructeurs

FIGUE offre un nombre minimal de constructeurs graphiques. Chaque constructeur prend une liste de boîtes ou des atomes et les formate selon un algorithme de formatage spécifique. Les constructeurs disponibles actuellement en FIGUE et les plus utilisés sont : *Atom*, *Horizontal*, *Vertical* et *Paragraphe*. *Atom* est un constructeur de base qui produit des feuilles de l'arbre de boîtes (atomes qui n'ont pas de fils). *Horizontal* formate en mode horizontal la liste de ses fils. Il prend en paramètre l'espace entre deux éléments. L'alignement se fait au niveau du point de sortie d'un élément et du point d'entrée de l'élément suivant (voir figure 4). *Vertical* dispose ses fils en mode vertical. Il prend en paramètre l'espace consécutif entre deux lignes et l'indentation (i.e. un décalage horizontal de tous les éléments par rapport au premier). *Paragraphe* met ses éléments en mode horizontal tant qu'il reste assez de marge à droite de la page. Sinon il retourne à la ligne et dispose à nouveau ses éléments horizontalement.

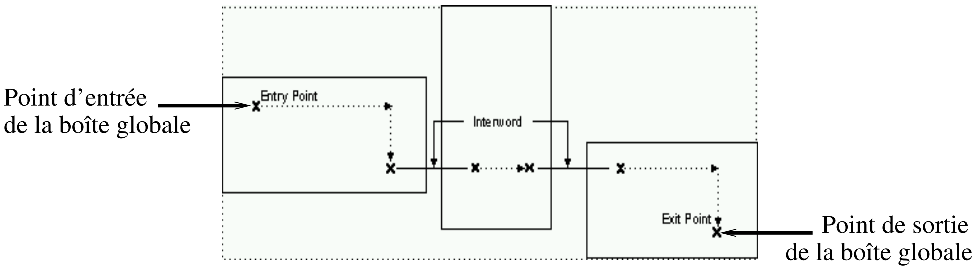


FIG. 4 – *Le constructeur Horizontal.*

En utilisant ces constructeurs de base, nous arrivons seulement à gérer une édition linéaire des documents (comme des programmes). Ce résultat n'est pas suffisant pour manipuler des formules mathématiques qui sont naturellement bidimensionnelles (intégrale, matrice, racine carrée,...). Utiliser l'extensibilité de FIGUE s'avère nécessaire pour introduire de nouveaux constructeurs pour les formules mathématiques.

3.6 Extensibilité de FIGUE pour les formules mathématiques

FIGUE possède une architecture générique facile à étendre. Il permet d'ajouter facilement de nouveaux constructeurs graphiques. En se basant sur son noyau de base (les constructeurs *Horizontal*, *Vertical*, ...), nous avons introduit de nouveaux constructeurs dans le but d'offrir la possibilité de manipuler dynamiquement des objets ayant une structure plus complexe (par exemple les formules mathématiques) pour produire une version de FIGUE plus puissante pour la gestion de documents contenant à la fois du texte et des formules mathématiques.

Pour ajouter un nouveau constructeur dans FIGUE, il faut d'abord définir comment il disposera ses fils par rapport à son origine (l'algorithme de formatage dépend de la nature du constructeur). Ensuite, il faut déterminer l'alignement de sa boîte englobante avec l'ensemble (ses points d'entrée et de sortie - voir figure 4). Finalement, il faut déterminer son algorithme incrémental d'affichage qui consiste à dessiner ses fils et éventuellement dessiner de nouveaux symboles.

Parmi les difficultés soulevées lors de l'introduction des constructeurs de formules mathématiques, nous avons rencontré le problème du dessin des délimiteurs (tels que $($, $\int$, $\{$). Les délimiteurs sont des symboles mathématiques de taille variable délimitant des expressions mathématiques plus ou moins grandes. De plus, le dessin de ces symboles doit tenir compte de leur taille et de leur contexte graphique [1]. Pour les dessiner, nous nous sommes inspirés du système METAFONT en LaTeX [10] [11] et des polices *outlines* [18] qui se basent sur les courbes de Bézier[3]. L'algorithme de dessin de chaque symbole calcule l'épaisseur et les paramètres des courbes représentant le contour du caractère en fonction de la taille de la police utilisée et de la hauteur du symbole. Ces symboles mathématiques de taille variable sont gérés au cas par cas. Arriver à trouver un algorithme général pour résoudre les problèmes du dessin de ces symboles demande d'étudier en profondeur les règles typographiques. Pour l'instant, le résultat obtenu pour l'affichage de ces symboles est largement satisfaisant en attendant la sortie d'une police vectorielle appropriée.

D'autre part, nous avons introduit en FIGUE des constructeurs mathématiques (*Racine*, *Puissance*, *Matrice*,...) qui formatent dynamiquement la formule mathématique correspondante, déterminent son alignement avec l'ensemble et au besoin dessinent les symboles nécessaires. Par exemple, le constructeur *Racine* dessine le signe racine autour sa première boîte-fille en tenant compte de sa taille (voir figure 5) et dispose son deuxième fils de façon à avoir une racine n-ième.

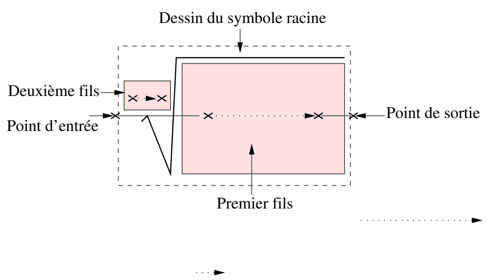


FIG. 5 – *Le formatage de la racine n-ième.*

L'algorithme de formatage et d'affichage dépend de la nature du constructeur mathématique. Par exemple, la disposition correcte des éléments d'une matrice requiert un algorithme en plusieurs passes[4]. L'incrémentalité dépend aussi de la sémantique de chaque constructeur (quels sont les éléments à reformater si on change un élément de la formule). Un soin particulier est accordé à l'efficacité des algorithmes de formatage et d'incrementalité pour les formules mathématiques pour assurer des délais d'affichage compatibles avec les besoins d'une interface.

La manipulation des objets mathématiques soulève plusieurs problèmes [13] [2], par exemple, la complexité et la diversité des règles typographiques, l'affichage des grandes formules, la sélection de sous-expressions et l'ambiguïté des notations mathématiques. Dans la suite de notre travail, nous espérons proposer des solutions intéressantes pour

3. Ces symboles peuvent être des courbes quelconques, par exemple des polynômes du troisième degré (cubiques) dont les courbes de Bézier sont un cas particulier.

4. La position d'un terme de la matrice n'est connu qu'a près avoir déterminé, par une recherche plus globale, la largeur de chaque colonne de la matrice et la hauteur de chaque ligne.

ces problèmes.

4 Conclusion

FIGUE est un module de formatage et d'affichage 2D conçu pour le développement d'outils WYSIWYG. La nouvelle version de FIGUE, réécrite totalement en JAVA, forme un module indépendant et portable. Actuellement, nous l'étendons pour manipuler et afficher des objets plus structurés, en particulier les formules mathématiques.

Nous fixons comme objectif, en se basant sur le noyau de base de FIGUE, de développer un éditeur pour manipuler des objets structurés pour le Web, dans le genre de AMAYA. L'avantage de notre approche est d'associer les objets à leur sens sémantique. Elle offre alors la possibilité de faire des traitements sur les objets affichés hors du contexte du document contrairement à l'approche de la plupart des éditeurs qui manipulent les objets en tant que texte. Par exemple, il serait possible de simplifier une formule mathématique d'un document.

Dans la version actuelle de PCOQ, Pottier [16] travaille sur la mise en place d'explication des preuves en langue naturelle. Aujourd'hui, il existe des explications en langue française et anglaise. Dans l'avenir, nous envisageons d'expérimenter l'implémentation des explications en arabe, ce qui permettrait de tester FIGUE pour écrire de droite à gauche et même mélanger l'affichage droite-gauche (texte arabe) et l'affichage gauche-droite (mathématiques).

Références

[1] J. André, I. Vatton. Dynamic Optical Scaling and Variable Sized Characters. Electronic Publishing – Origination, Dissemination and Design, vol. 7, num. 4, pp. 231-250,December 1994.

[2] O. Arsac. *Interfaces homme machine pour le calcul formel.* PhD thesis, Université de Nice Sophia Antipolis, Juillet 1997.

[3] Y. Bertot. The ctcoq system: Design and architecture. Technical Report RR-3540, Inria, Institut National de Recherche en Informatique et en Automatique.

[4] Y. Bertot, G. Kahn, and L. Thery. Proof by pointing. *Lecture Notes in Computer Science*, 789:141–??, 1994.

[5] Y. Bertot. Direct manipulation of Algebraic Formulae in Interactive Proof Systems *Workshop on User-Interfaces for Theorem Provers*, September 1997. ftp://babar.inria.fr/pub/croap/bertot/uitp-97.ps.gz

[6] P. Borras, D. Clement, T. Despeyroux, J. Incerpi, G. Kahn, B. Lang, and V. Pascual. CENTAUR: The system. Research Report 777, INRIA, Rocquencourt, France, December 1987.

[7] Gerard Huet, Gilles Kahn, and Christine Paulin-Mohring. The coq proof assistant : A tutorial : Version 6.1. Technical Report RT-0204, Inria, Institut National de Recherche en Informatique et en Automatique.

[8] I. Jacobs and L. Rideau-Gallot. The ppml manual. Technical report, Inria, Institut National de Recherche en Informatique et en Automatique, August 1994. ftp://ftp-sop.inria.fr/lemme/papers/ppml.ps

[9] N. Kajler. User interfaces for symbolic computation: a case study. In *Proceedings of the 6th Annual Symposium on User Interface Software and Technology*, pages 1–10, New York, NY, USA, November 1993. ACM Press.

[10] D. Knuth. *Computers and Typesetting.* Addison-Wesley, Reading, 1986.

[11] D. Knuth. Typesetting *Concrete Mathematics*. *TUGboat*, 10(1):31–36, April 1989.

[12] D. Knuth. *The TeX-Book (revised).* Addison Wesley, 1990.

[13] N. Soiffer. The design of a user interface for computer algebra systems. Technical Report CSD-91-626, University of California, Berkeley.

[14] http://www-sop.inria.fr/croap/figue/

[15] http://www-sop.inria.fr/croap/aioli/

[16] http://www-sop.inria.fr/lemme/pcoq/pcoq-fra.html

[17] http://www.w3.org/Amaya/

[18] Roger D. Hersch. Visual and Technical Aspects of Type, Cambridge University Press, New York, 1993.